Package 'hca'

December 30, 2024

Title Exploring the Human Cell Atlas Data Coordinating Platform

Version 1.15.0

Description This package provides users with the ability to query the Human Cell Atlas data repository for single-cell experiment data. The `projects()`, `files()`, `samples()` and `bundles()` functions retrieve summary information on each of these indexes; corresponding `*_details()` are available for individual entries of each index. File-based resources can be downloaded using `files_download()`. Advanced use of the package allows the user to page through large result sets, and to flexibly query the 'list-of-lists' structure representing query responses.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Depends R (>= 4.1)

Imports httr, jsonlite, dplyr, tibble, tidyr, readr, BiocFileCache, tools, utils, digest, shiny, miniUI, DT

Suggests LoomExperiment, SummarizedExperiment, SingleCellExperiment, S4Vectors, methods, testthat (>= 3.0.0), knitr, rmarkdown, BiocStyle

Config/testthat/edition 3

VignetteBuilder knitr

biocViews Software, SingleCell

git_url https://git.bioconductor.org/packages/hca

git_branch devel

git_last_commit 2bb75df

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-30

Author Maya McDaniel [aut],
Martin Morgan [aut, cre] (ORCID:
<https://orcid.org/0000-0002-5874-8148>),
Kayla Interdonato [ctb]

Maintainer Martin Morgan <martin.morgan@roswellpark.org>

Contents

`.details` *Single Entity Details*

Description

Single Entity Details

Usage

```
.details(
  uuid = character(),
  catalog = NULL,
  view = c("projects", "files", "samples", "bundles")
)
```

Arguments

`uuid`	character() unique *_id
`catalog`	character(1) source of data. Use `catalogs()` for possible values.
`view`	character() type of entity i.e. project, file, sample, or bundle

Value

list-of-lists containing relevant details about the project, file, sample, or bundle

---

bundles *HCA Bundle Querying*

Description

`bundles()` takes a list of user provided project titles to be used to query the HCA API for information about available bundles.

`bundles_detail()` takes a unique bundle_id and catalog for the bundle, and returns details about the specified bundle as a list-of-lists

Usage

```
bundles(
  filters = NULL,
  size = 100L,
  sort = "projectTitle",
  order = c("asc", "desc"),
  catalog = NULL,
  as = c("tibble", "lol", "list", "tibble_expanded"),
  columns = bundles_default_columns("character")
)

bundles_facets(facet = character(), catalog = NULL)

bundles_default_columns(as = c("tibble", "character"))

bundles_detail(uuid, catalog = NULL)
```

Arguments

`filters`	filter object created by `filters()`, or `NULL` (default; all projects).
`size`	integer(1) maximum number of results to return; default: all projects matching `filter`. The default (10000) is meant to be large enough to return all results.
`sort`	character(1) project facet (see `facet_options()`) to sort result; default: `"projectTitle"`.
`order`	character(1) sort order. One of `"asc"` (ascending) or `"desc"` (descending).
`catalog`	character(1) source of data. Use `catalogs()` for possible values.
`as`	character(1) return format. One of `"tibble"` (default), `"lol"`, `"list"`, or `"tibble_expanded"`, as described in the Details and Value sections of `?projects`.
`columns`	named character() indicating the paths to be used for parsing the 'lol' returned from the HCA to a tibble. The names of `columns` are used as column names in the returned tibble. If the columns are unnamed, a name is derived from the elements of `path` by removing `hits[*]` and all `[*]`, e.g., a path `hits[*].donorOrganisms[*].biological` is given the name `donorOrganisms.biologicalSex`.

facet	character() of valid facet names. Summary results (see 'Value', below) are returned when missing or length greater than 1; details are returned when a single facet is specified.
uuid	character() unique identifier (e.g., projectId) of the object.

Value

bundles_detail() returns a list-of-lists containing relevant details about the bundle

Examples

```
title <- paste(
    "Tabula Muris: Transcriptomic characterization of 20 organs and",
    "tissues from Mus musculus at single cell resolution"
)
filters <- filters( projectTitle = list(is = title) )
bundles(filters = filters)

bundles_facets()

bundle <- bundles(size = 1, as = "list")
bundle_uuid <- bundle[["hits"]][[1]][["entryId"]]
bundles_detail(uuid = bundle_uuid) |> lol() |>
lol_filter(is_leaf) |> print(n = Inf)
```

---

catalogs *Catalogs Available in the HCA*

Description

catalogs() queries the API for all available project catalogs

Usage

```
catalogs(catalog = NULL)
```

Arguments

catalog	character(1) default catalog. When missing or NULL, the catalog defined by the Human Cell Atlas API is used; this is usually the most recently available catalog. Providing a non-null argument changes the default globally; restore default order by explicitly defining the arugment catalog = NULL.

Value

character() vector of available catalogs. The first is the default, defined by the API or by the user with argument catalog.

Examples

```
catalogs()
```

`files` *HCA File Querying*

Description

`files()` takes a list of user provided project titles to be used to query the HCA API for information about available files.

`files_download()` takes a tibble of files and a directory location as arguments to download the files of the tibble into the specified directory.

`files_detail()` takes a unique file_id and catalog for the file, and returns details about the specified file as a list-of-lists

`files_cache()` is the default location of the cache of downloaded files.

Usage

```
files(
  filters = NULL,
  size = 1000L,
  sort = "projectTitle",
  order = c("asc", "desc"),
  catalog = NULL,
  as = c("tibble", "lol", "list", "tibble_expanded"),
  columns = files_default_columns("character")
)

files_default_columns(as = c("tibble", "character"))

files_download(tbl, destination = NULL)

files_facets(facet = character(), catalog = NULL)

files_detail(uuid, catalog = NULL)

files_cache(create = FALSE)
```

Arguments

`filters`	filter object created by `filters()`, or `NULL` (default; all projects).
`size`	integer(1) maximum number of results to return; default: all projects matching `filter`. The default (10000) is meant to be large enough to return all results.
`sort`	character(1) project facet (see `facet_options()`) to sort result; default: `"projectTitle"`.

order	character(1) sort order. One of "asc" (ascending) or "desc" (descending).
catalog	character(1) source of data. Use catalogs() for possible values.
as	character(1) return format. One of "tibble" (default), "lol", "list", or "tibble_expanded", as described in the Details and Value sections of ?projects.
columns	named character() indicating the paths to be used for parsing the 'lol' returned from the HCA to a tibble. The names of columns are used as column names in the returned tibble. If the columns are unnamed, a name is derived from the elements of path by removing hits[*] and all [*], e.g., a path hits[*].donorOrganisms[*].biological is given the name donorOrganisms.biologicalSex.
tbl	tibble of files (result of files())
destination	character() vector name of temporary directory to use for file downloads, or NULL
facet	character() of valid facet names. Summary results (see 'Value', below) are returned when missing or length greater than 1; details are returned when a single facet is specified.
uuid	character() unique identifier (e.g., projectId) of the object.
create	logical(1) create the default cache location, if it does not yet exist.

Details

files_cache() can be useful when it is necessary to 'clean up' the cache, e.g., BiocFileCache::cleanbfc() or more dramatically unlink(files_cache(), recursive = TRUE).

Value

files_download() returns a character() vector of file destinations

files_detail() returns a list-of-lists containing relevant details about the file.

files_cache() returns the path to the default cache. Use this as the cache= argument to BiocFileCache().

Examples

```
title <- paste(
    "Tabula Muris: Transcriptomic characterization of 20 organs and",
    "tissues from Mus musculus at single cell resolution"
)
filters <- filters( projectTitle = list(is = title) )
files(filters = filters)

files_filter <- filters(
    projectId = list(is = "cddab57b-6868-4be4-806f-395ed9dd635a"),
    fileFormat = list(is = "loom")
)
files_tbl <- files(filter = files_filter)
## Not run: files_download(files_tbl, destination = tempdir())
files_facets()
files_facets("fileFormat")
```

```
file <- files(size = 1, as = "list")
file_uuid <- file[["hits"]][[1]][["entryId"]]
files_detail(uuid = file_uuid)

files_cache(create = FALSE)
```

---

filters *HCA Filter Construction*

Description

`facet_options()` returns a character vector of possible facets to use during filtering.

`filters()` takes user input to be used as query filters. Each named argument is a list with a name specifying a verb (e.g., `"is"`) and a character vector of allowed values, as in the examples. This input is then validated, reformatted to JSON, and encoded into a properly formatted URL.

Usage

```
facet_options()

filters(...)

## S3 method for class 'filters'
length(x)

## S3 method for class 'filters'
print(x, ...)
```

Arguments

`...`	named arguments, each of which is a `list()` specifying a query facet and its corresponding value to be used in the query
`x`	for `length()` and `print()`, an object of class `filters`.

Value

`facet_options()` returns a vector of all permissible query facets for the HCA api.

`filters()` returns a `filters` object representing validated filters in a format suitable for use in `projects()` and related functions.

Examples

```
facet_options()

filters()

filters(
    organ = list(is = "pancreas")
```

```
)

filters(
    organ = list(is = "pancreas"),
    genusSpecies = list(is = "Homo sapiens")
)

filters(
    fileFormat = list(is = c("fastq", "fastq.gz"))
)
```

---

hca_next *Page through HCA results*

Description

hca_next() retrieves the next 'page' of results from a query of projects(), samples(), files(), or bundles().

hca_prev() returns the previous 'page' of results.

Usage

```
hca_next(x, size)

hca_prev(x, size)
```

Arguments

x	a 'tibble' or 'lol' object returned by projects(), samples(), files(), or bundles().
size	the (non-negative integer) number of elements to retrieve in the page request. The default is the number of elements requested in x.

Value

hca_next() returns the next page of results as a 'tibble' or 'lol'

hcl_prev() returns the previous page of results.

Examples

```
files <- files(size = 5)         # results 1-5, as a tibble

next_files <- hca_next(files)    # results 6-10
next_files

hca_prev(next_files)             # previous results, i.e., files 1-5
```

hca_next.list_hca *'list' representation of HCA query results*

Description

projects(), samples(), files() and bundles() return results for the number of records indicated by the size= argument. Use as = "list" to return results as a "list_hca" list.

hca_next() returns a list containing the next 'page' of results.

hca_prev() returns a list containing the previous 'page' of results.

Usage

```
## S3 method for class 'list_hca'
hca_next(x, size)

## S3 method for class 'list_hca'
hca_prev(x, size)
```

Arguments

x	a 'list' returned by projects(), samples(), files(), or bundles().
size	the (non-negative integer) number of elements to retrieve in the page request. The default is the number of elements requested in x.

Value

hca_next() returns a list containing the next 'page' of results.

hca_prev() returns a list containing the previous 'page' of results.

Examples

```
projects <- projects(size = 5, as = "list") # projects 1-5
next_projects <- hca_next(projects)         # projects 6-10

hca_prev(next_projects)            # projects 1-5
```

hca_next.tbl_hca *'tibble' representation of HCA query results*

Description

projects(), samples(), files(), and bundles() return, by default, a 'tibble' representation of the query.

hca_next() returns the next 'page' of results, if available.

hca_prev() returns the previous 'page' of results.

Usage

```
## S3 method for class 'tbl_hca'
hca_next(x, size)

## S3 method for class 'tbl_hca'
hca_prev(x, size)
```

Arguments

x	a 'tibble' returned by projects(), samples(), files(), or bundles().
size	the (non-negative integer) number of elements to retrieve in the page request. The default is the number of elements requested in x.

Value

hca_next() returns a tibble, with the same columns as x, containing the next 'page' of results.

hca_prev() returns a tibble with the same columns as x, containing the previous 'page' of results.

Examples

```
projects <- projects(size = 5)      # projects 1-5
next_projects <- hca_next(projects) # projects 6-10

hca_prev(next_projects)             # projects 1-5
```

hca_view *View and select table rows interactively*

Description

View and select table rows interactively

Usage

```
hca_view(tbl)
```

Arguments

tbl	a 'tibble' of `projects()`, `samples()`, `bundles()`, or `files()`.

Value

`hca_view()` returns a tibble filtered to reflect the rows selected in the interface.

Examples

```
if (interactive()) {
    p <- projects(size = 100)
    p1 <- hca_view(p)  # interactive table browser; filtered results
}
```

lol *Representing and manipulating list-of-list data structures.*

Description

`lol()` constructs an indexed representation of an R 'list-of-lists', typically from JSON queries. The object is conveniently manipulated by other functions on this page to filter and select subsets of the structure, and to pull individual paths from across the list-of-lists.

`lol_filter()` filters available paths based on selections in `...`, e.g., `n` (number of matching elements) or `is_leaf` (is the element a 'leaf' in the list-of-lists representation?).

`lol_lpull()` returns a list containing elements corresponding to a single `path`.

`lol_pull()` tries to simplify the list-of-lists structure returned by `lol_lpull()` to a vector.

`lol_path()` returns a tibble representing the paths through the list-of-lists, without the underlying list-of-list data.

`as.list()` returns a list-of-lists representation of the data returned by `projects()`, etc.

`hca_next()` returns the next 'page' of results, if available.

`hca_prev()` returns the previous 'page' of results.

`lol_hits_lpull()` and `lol_hits_pull()` are variants of `lol_lpull()` and `lol_pull()` that retain the original geometry of `hits[*]`, even when the mapping between `hits[*]` and `path` is not 1:1.

Usage

```
lol(x = list())

lol_select(x, path = character())

lol_filter(x, ...)

lol_lpull(x, path)

lol_pull(x, path)

lol_path(x)

## S3 method for class 'lol'
as.list(x, ...)

## S3 method for class 'lol'
print(x, ...)

## S3 method for class 'lol_hca'
hca_next(x, size)

## S3 method for class 'lol_hca'
hca_prev(x, size)

lol_hits_lpull(x, path)

lol_hits_pull(x, path)
```

Arguments

x	a 'list-of-lists' returned by `projects()`, `samples()`, `files()`, or `bundles()`
path	character(1) from the tibble returned by `lol_path(x)`.
...	for `lol_filter()`, named filter expressions evaluating to a logical vector with length equal to the number of rows in `lol_path()`.
size	the (non-negative integer) number of elements to retrieve in the page request. The default is the number of elements requested in `x`.

Value

`lol()` returns a representation of the list-of-lists. The list has been processed to a dictionary with entries to all paths through the list, as well as a tibble summarizing the path, number of occurrences, and leaf status of each unique path.

`lol_select()` returns an object of class `"lol"` subset to contain just the elements matching `path` as 'top-level' elements of the list-of-lists.

`lol_filter()` returns an object of class `lol`, filtered to contain elements consistent with the filter criteria.

`lol_lpull()` returns a list, where each element corresponds to an element found at `path` in the list-of-lists structure `x`.

`lol_pull()` returns an unnamed vector of elements matching `key`.

`hca_next()` returns a list-of-lists containing the next 'page' of results.

`hca_prev()` returns a tibble with the same columns as `x`, containing the previous 'page' of results.

Examples

```
plol <- projects(size = 5, as = "lol")
plol

plol |> lol_select("hits[*].projects[*]")

plol |>
   lol_select("hits[*].projects[*]") |>
   lol_filter(n == 44, is_leaf)

plol |>
    lol_pull("hits[*].entryId") |>
    head()

plol |> lol_path()

projects <- projects(size = 5, as = "lol")     # projects 1-5
next_projects <- hca_next(projects)            # projects 6-10

hca_prev(next_projects)                        # projects 1-5
```

`manifest` *HCA File Querying*

Description

`manifest()` takes a list of user provided project titles to be used to query the HCA API for information about available manifest files.

`manifest_cache()` is the default location of the cache of downloaded manifest.

Usage

```
manifest(filters = NULL, catalog = NULL, update_cache = FALSE)

manifest_cache(create = FALSE)
```

Arguments

filters	hca filter object
catalog	character() name of catalog
update_cache	logical(1) when TRUE, update an existing cached resource by querying the HCA data server.
create	logical(1) create the default cache location, if it does not yet exist.

Details

manifest_cache() can be useful when it is necessary to 'clean up' the cache, e.g., BiocFileCache::cleanbfc() or more dramatically unlink(manifest_cache(), recursive = TRUE).

Value

manifest_cache() returns the path to the default cache. Use this as the cache= argument to BiocFileCache().

Examples

```
manifest_filter <- hca::filters(
    projectId = list(is = "4a95101c-9ffc-4f30-a809-f04518a23803"),
    fileFormat = list(is = "loom"),
    workflow = list(is = c("optimus_v4.2.2", "optimus_v4.2.3"))
)
## Not run:
result <- manifest(manifest_filter)
result

## End(Not run)
manifest_cache(create = FALSE)
```

---

manifest_url_generator

Internal functions used by manifest()

Description

manifest_url_generator() takes a filter object with criteria for the query and the catalog to search within.

manifest_uuid_constructor() takes a filter object with criteria for the query and the catalog to search within.

Usage

```
manifest_url_generator(manifest_filter, catalog)

manifest_uuid_constructor(manifest_filter, catalog)
```

Arguments

manifest_filter	hca filter object
catalog	character() name of catalog

Value

manifest_url_generator() returns a url string to be used in an API call

manifest_uuid_constructor() returns a url string to be used in an API call

Examples

```
hca:::manifest_url_generator(filters(), catalogs()[1])
hca:::manifest_uuid_constructor(filters(), catalogs()[1])
```

---

optimus_loom_annotation

HCA loom file annotation

Description

optimus_loom_annotation() takes the file path location of a .loom file generated by the Optimus pipeline, for which additional data will be extracted from the appropriate manifest. The .loom file will be imported as a LoomExperiment object, and the additional manifest information will be added to the object for return.

Usage

```
optimus_loom_annotation(loom, catalog = NULL)

## S3 method for class 'character'
optimus_loom_annotation(loom, catalog = NULL)

## S3 method for class 'LoomExperiment'
optimus_loom_annotation(loom, catalog = NULL)
```

Arguments

loom	Either a character(1) file path to a loom file on user's system, or a loom file obtained from the HCA and imported into R using LoomExperiment::import().
catalog	character() HCA catalog from which the .loom file originated.

Value

A 'LoomExperiment' object annotated with additional metadata() and colData() derived from the manifest file describing samples in the object.

See Also

manifest() and related functions for working with data returned from the */manifest/* HCA API endpoints.

---

projects *HCA Project Querying*

Description

projects() takes user input to be used to query the HCA API for information about available projects.

projects_facets() summarizes facets and terms used by all records in the projects index. *_columns() returns a tibble or named character vector describing the content of the tibble returned by projects(), files(), samples(), or bundles().

projects_detail() takes a unique project_id and catalog for the project, and returns details about the specified project as a list-of-lists

See project_information() and project_title() to easily summarize a project from its project id.

Usage

```
projects(
  filters = NULL,
  size = 1000L,
  sort = "projectTitle",
  order = c("asc", "desc"),
  catalog = NULL,
  as = c("tibble", "lol", "list", "tibble_expanded"),
  columns = projects_default_columns("character")
)

projects_facets(facet = character(), catalog = NULL)

projects_default_columns(as = c("tibble", "character"))

projects_detail(uuid, catalog = NULL)
```

Arguments

filters	filter object created by filters(), or NULL (default; all projects).
size	integer(1) maximum number of results to return; default: all projects matching filter. The default (10000) is meant to be large enough to return all results.
sort	character(1) project facet (see facet_options()) to sort result; default: "projectTitle".
order	character(1) sort order. One of "asc" (ascending) or "desc" (descending).

catalog	character(1) source of data. Use catalogs() for possible values.
as	character(1) return format. One of "tibble" (default), "lol", "list", or "tibble_expanded", as described in the Details and Value sections of ?projects.
columns	named character() indicating the paths to be used for parsing the 'lol' returned from the HCA to a tibble. The names of columns are used as column names in the returned tibble. If the columns are unnamed, a name is derived from the elements of path by removing hits[*] and all [*], e.g., a path hits[*].donorOrganisms[*].biological is given the name donorOrganisms.biologicalSex.
facet	character() of valid facet names. Summary results (see 'Value', below) are returned when missing or length greater than 1; details are returned when a single facet is specified.
uuid	character() unique identifier (e.g., projectId) of the object.

Details

The as argument determines the object returned by the function. Possible values are:

- "tibble" (default) A tibble (data.frame) summarizing essential elements of projects, samples, bundles, or files.
- "lol" A 'list-of-lists' representation of the JSON returned by the query as a 'list-of-lists' data structure, indexed and presented to enable convenient filtering, selection, and extraction. See ?lol.
- "list" An R list (typically, highly recursive) containing detailed project information, constructed from the JSON response to the original query.
- "tibble_expanded" A tibble (data.frame) containing (almost) all information for each project, sample, bundle, or file. The exception is user-contributed matrices present in projects() records; these must be accessed using the "lol" format to extract specific paths as a standard "tibble".

Value

When as = "tibble" or as = "tibble_expanded", a tibble with each row representing an HCA object (project, sample, bundle, or file, depending on the function invoked), and columns summarizing the object. "tibble_expanded" columns contains almost all information about the object, except as noted in the Details section.

When as = "lol", a list-of-lists data structure representing detailed information on each object.

When as = "list", projects() returns an R list, typically containing other lists or atomic vectors, representing detailed information on each project.

projects_facets() invoked with no facet= argument returns a tibble summarizing terms available as projects() return values, and for use in filters. The tibble contains columns

- facet: the name of the facet.
- n_terms: the number of distinct values the facet can take.
- n_values: the number of occurrences of the facet term in the entire catalog.

`projects_facets()` invoked with a scalar value for `facet=` returns a tibble summarizing terms used in the facet, and the number of occurrences of the term in the entire catalog.
`*_columns()` returns a tibble with column `name` containing the column name used in the tibble returned by `projects()`, `files()`, `samples()`, or `bundles()`, and `path` the path (see `lol_hits()`) to the data in the list-of-lists by the same functions when `as = "lol"`. When `as = "character"`, the return value is a named list with paths as elements and abbreviations as names.

list-of-lists containing relevant details about the project.

See Also

`project_information()` and `project_title()` to easily summarize a project from its project id.

`lol()` and other `lol_*()` functions for working with the list-of-list data structure returned when `as = "lol"`.

Examples

```
projects(filters(), size = 100)

projects_facets()
projects_facets("genusSpecies")

projects_default_columns()

project <- projects(size = 1, as = "list")
project_uuid <- project[["hits"]][[1]][["entryId"]]
projects_detail(uuid = project_uuid)
```

---

project_information *Project Summaries from Project IDs*

Description

`project_information()` queries the HCA database for project title, description, contact, DOI, and publication URI.

`project_title()` returns the title of the project, cleaned to remove trailing trailing ..

`print.project_information()` formats the result of `project_information()` in a more legible manner.

Usage

```
project_information(project_id)

project_title(project_id)

## S3 method for class 'project_information'
print(x, ...)
```

Arguments

project_id	character(1) project identifier, e.g., "3c9d586e-bd26-4b46-8690-3faaa18ccf38".
x	an object of class project_information, the result of a call to project_information().
...	additional arguments, required to conform with the print generic but not used.

Value

project_information() returns a tibble with a single row, and columns containing information about the project. The tibble is of class project_information and is printed in an interactive session formatted so long columns, e.g., projectDescription, are more easily read.

project_title() returns a character(1) vector containing the project title.

print.project_information() is invoked automatically when the result of project_information() is displayed for it's side effect of displaying the object.

Examples

```
project_id <- "3c9d586e-bd26-4b46-8690-3faaa18ccf38"
project_information(project_id)

project_title(project_id)
```

---

samples *HCA Sample Querying*

Description

samples() takes a list of user provided project titles to be used to query the HCA API for information about available samples.

samples_detail() takes a unique sample_id and catalog for the sample, and returns details about the specified sample as a list-of-lists

Usage

```
samples(
  filters = NULL,
  size = 1000L,
  sort = "projectTitle",
  order = c("asc", "desc"),
  catalog = NULL,
  as = c("tibble", "lol", "list", "tibble_expanded"),
  columns = samples_default_columns("character")
)

samples_facets(facet = character(), catalog = NULL)
```

```
samples_default_columns(as = c("tibble", "character"))

samples_detail(uuid, catalog = NULL)
```

Arguments

filters	filter object created by `filters()`, or `NULL` (default; all projects).
size	integer(1) maximum number of results to return; default: all projects matching `filter`. The default (10000) is meant to be large enough to return all results.
sort	character(1) project facet (see `facet_options()`) to sort result; default: `"projectTitle"`.
order	character(1) sort order. One of `"asc"` (ascending) or `"desc"` (descending).
catalog	character(1) source of data. Use `catalogs()` for possible values.
as	character(1) return format. One of `"tibble"` (default), `"lol"`, `"list"`, or `"tibble_expanded"`, as described in the Details and Value sections of `?projects`.
columns	named character() indicating the paths to be used for parsing the 'lol' returned from the HCA to a tibble. The names of `columns` are used as column names in the returned tibble. If the columns are unnamed, a name is derived from the elements of `path` by removing `hits[*]` and all `[*]`, e.g., a path `hits[*].donorOrganisms[*].biological` is given the name `donorOrganisms.biologicalSex`.
facet	character() of valid facet names. Summary results (see 'Value', below) are returned when missing or length greater than 1; details are returned when a single facet is specified.
uuid	character() unique identifier (e.g., `projectId`) of the object.

Value

`samples_detail()` returns a list-of-lists containing relevant details about the sample

Examples

```
title <- paste(
    "Tabula Muris: Transcriptomic characterization of 20 organs and",
    "tissues from Mus musculus at single cell resolution"
)
filters <- filters( projectTitle = list(is = title) )
samples(filters = filters)

samples_facets()

sample <- samples(size = 1, as = "list")
sample_uuid <- sample[["hits"]][[1]][["entryId"]]
samples_detail(uuid = sample_uuid)
```

summary *Repository summary statistics*

Description

summary() provides numerical summaries of catalog content

Usage

```
summary(
  filters = NULL,
 type = c("overview", "fileTypeSummaries", "cellCountSummaries", "organTypes", "list"),
  catalog = NULL
)
```

Arguments

filters	filter object created by filters(), or NULL (default; all projects).
type	character(1) type of summary to return. Possible values include "overview", "fileTypeSummaries", "cellCountSummaries", "organType", and a "list" off all summary statistics.
catalog	character(1) source of data. Use catalogs() for possible values.

Value

summary() returns a tibble or (for type = "list") a list-of-lists of summary statistics.

Examples

```
hca::summary()

filter <- filters(
    organ = list(is = c("brain", "heart")),
    genusSpecies = list(is = "Homo sapiens")
)
hca::summary(filter)
hca::summary(filter, "fileTypeSummaries")
hca::summary(filter, "cellCountSummaries")
```

Index